

Honeyd: Simple, Cost-Effective Detection

Lance Spitzner 2003-04-30

Honeyd: Simple, Cost-Effective Detection

by *Lance Spitzner*, www.tracking-hackers.com

last updated April 30, 2003

This is the fourth article in an ongoing series examining honeyd. In previous installments, we have covered two different honeyd solutions: [Honeyd](#) and [Specter](#). Both honeyd are low-interaction production solutions; their purpose is to help protect organizations, as opposed to research honeyd, which are used to gather information. Production honeyd work by emulating a variety of services and operating systems. Honeyd, an OpenSource solution, is considered more powerful and flexible than Specter, but it is also more difficult to use. Specter, a commercially supported solution, is easier to use as it runs on Windows. In this paper we take a step back for a moment and discuss the value of honeyd technologies in general. Why would you want to deploy production honeyd in your organization? How can a honeyd help security professionals to do their job more effectively?

As you are about to find out, the answer is very simple: honeyd are a simple, cost-effective way to detect illicit, unauthorized activity. This article will examine the role of detection in the overall security strategy. It will then discuss some traditional detection approaches as well as some problems inherent in those approaches. It will then show how honeyd effectively overcome those problems, thereby strengthening the detection component of the security strategy.

What is Detection?

Many organizations approach security as three different layers: prevention, detection, and response. Prevention is the process of keeping the bad guys out. Detection is consists of identifying there is a failure in prevention and alerting systems administrators accordingly. Finally, response is the way in which the organization reacts to the security incident. Of the three, I consider detection to be the most critical. No matter what preventative technologies, processes, or programs are in place, sooner or later there will be a failure. The simple reason for this is that security is based on humans, and humans make mistakes. When such a failure happens, it is critical that security personnel quickly identify and react to it.

The advantages to effective detection are twofold. First, by quickly identifying unauthorized activity, you may be able to stop an attack before it happens. For example, if you detect someone on your internal network scanning for open file shares, you can identify and stop that person before they find any files they should not have access to. Second, timely detection can be used to mitigate a successful compromise. If you quickly determine that a system or resource has been taken over, you can isolate and recover the system. However, if an attacker takes over your mail server, and its not discovered for over a month, extensive damage can be done as they intercept and monitor your communications.

Traditional Detection Approaches (and Problems)

You would think that detecting and alerting on unauthorized activity would be easy. Unfortunately, it's not, as many readers will already know. Traditionally, one of the most common detection technologies has been network intrusion detection systems (NIDS). These systems work by passively monitoring network traffic for suspicious or unauthorized activity. When these systems identify such activity they generate an alert.

The trick to NIDS is defining how it identifies suspicious or unauthorized activity. There are many ways to do this; however, the two most common approaches are is rules based and anomaly based. Rules-based NIDS is based on a series patterns or signatures, which are essentially strings of code that are known to be indicative of malicious traffic. If a packet (or collection of packets) trying to enter the network contains these known patterns, it assumes unauthorized activity is occurring and an alert is generated. For example, below is an IDS rule from the OpenSource solution [Snort](#). This particular rule is designed to capture attacks against an FTP server. It looks for packets going to port 21 that contain "5057 440A 2F69". This content is given as a hexadecimal value. Whenever that content is matched, an alert is generated.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21
(msg:"FTP EXPLOIT overflow"; flow:to_server,established;
content:"|5057 440A 2F69|"; classtype:attempted-admin;
sid:340; rev:3;)
```

Unfortunately, NIDS are notorious for generating false positives, or alerting for attacks on traffic that is, in fact, benign. For example, when someone uploads an image to a file server, part of the image may match the content, even though the traffic is legitimate, falsely generating an alert. When a rules-based NIDS is deployed, it can take an organization several

weeks to tweak the rule-base, identifying which rules falsely trigger alerts. The reduction of false positives is critical. If too many are generated, organizations will ignore their detection mechanisms, just as most people simply ignore when car alarms go off.

While false positives are a problem with NIDS, so is the problem of updating. As new attacks are identified, new rules have to be added to the database or the NIDS will fail to identify the new attacks. This process of updating rules never ends, just as the development of new attacks never ends.

Anomaly-based NIDS is different. It does not work on rules; instead, anomaly based technologies attempt to learn what is normal network behavior, then triggers an alert on anything considered abnormal, to include an attack. The challenge to anomaly-based detection is defining what is normal. If a network does not change over a long period of time, then it becomes relatively simple to learn what normal is. However, new applications and technologies are constantly being added to networks, so change is often the norm. Chat programs, peer-to-peer networks, new implementations of existing protocols, the dynamic nature of IP communications, all of this makes it extremely difficult to establish a metric by which to establish normal network behavior.

The advantages rule-based and anomaly-based technologies share are they can be quickly deployed and monitor most systems quite comprehensively. By passively monitoring all systems, almost all network activity can be analyzed for suspicious activity. However, most NIDS face the same challenges:

- **Data Overload** - network intrusion detection systems tend to generate an extremely large volume of alerts. This volume makes it time consuming, resource intensive, and costly to analyze and review all data generated. For example, I know of organizations that generate over 100,000 alerts a day. This makes NIDS very costly to scale. They also require extensive manpower to analyze all of this information.
- **False Positives** - Of all the disadvantages of NIDS, this is one of the greatest, false alerts. Many NIDS have difficulty distinguishing between legitimate activities and malicious traffic that bear similarities. For instance, a BugTraq post with example exploit code may be interpreted by an NIDS as a buffer overflow because the sample code matches a specific rule or pattern. In another instance, anomaly-based detection technology may mistake new traffic introduced to your network based on the new Lotus Notes server you are using for an attack based on the fact that it is not normal traffic.

Even for organizations that have spent extensive time tuning their systems, false alerts are still a common problem. This can quickly degenerate into the 'little boy who cried wolf' scenario. If the IDS is repeatedly generating false positives, administrators begin to ignore the technology they are using for detection.

- **False Negatives** - Just as NIDS can often generate false alerts, they can also fail to alert, especially for new attacks. Attackers may develop new tools or methods that are designed to bypass NIDS (such as [AMDmutate](#), or new attacks that have never been captured before. This can leave organizations vulnerable to new attacks and techniques.
- **Resources** - NIDS require resource-intensive hardware to keep up with organization's activity and traffic. The faster your network and the more data you have, the bigger your NIDS will have to be to keep up. In addition to this, it will require large databases to store all of the data. This is becoming more of a problem as networks migrate from 10/100 Megabit to Gigabit networks.
- **Encryption** - More and more organizations are moving to encryption, in which all of the data is encrypted by methods such as SSH, SSL, and IPsec. This move is based on both best practices and regulation (such as HIPAA). But this very same technology can also blind us to what is happening on our networks. How can a NIDS detect an attack, when all it can see is an encrypted SSL stream on the wire. The very same technologies we are using to protect our networks can paradoxically blind our detection technologies.
- **IPv6** - IPv6 is the new protocol version for the Internet Protocol. Not widely adopted, it is mainly used in Asian countries, such as Japan. Most NIDS technologies are not capable of analyzing or understanding IPv6 packets. Even in strictly IPv4 networks, this is a problem, as attacks can enable IPv6 tunneling within IPv4, blinding detection technologies.

Honeybots as a Detection Solution

Honeybots are a relatively new security technology whose real value lies in being probed, attacked, or compromised so that the actions of the intruders can be observed, analyzed and understood. The concept is simple: they do not have any production purpose, there is no authorized interaction with them, so any interaction with a honeybot is most likely a probe, scan or attack. In general there are two types of honeybots: research and production. In the past, most attention has been given to research honeybots, such as the work done by the

[Honeynet Project](#). However, very little attention has been given to the capabilities of production honeypots. These are used to directly secure your organization. There is tremendous value in such honeypots, especially for detection.

The reason for this detection value lies in the way that honeypots work. As stated, since a honeypot has no production activity, no authorized, legitimate interactions will take place on it. Anytime anything or anyone is interacting with the honeypot, it is most likely indicative of unauthorized or malicious activity. This concept is extremely simple, but extremely effective. Honeypots are in many ways the very opposite of NIDS and other traditional detection technologies. Where NIDS fail honeypots can excel. Let's take a look at the disadvantages of NIDS, and see how honeypots compare.

- **Small Data Sets** - Honeypots only collect data when someone or something is interacting with them. Organizations that may log thousands of alerts a day may only log a hundred alerts with honeypots. This makes the data honeypots collect much easier to manage and analyze.
- **Reduced False Positives** - Honeypots dramatically reduce false positives. Any activity with honeypots is by definition unauthorized, making it extremely effective at detecting attacks. This allows organizations to quickly and easily reduce, if not eliminate, false alerts, allowing organizations to focus on other security priorities, such as patching.
- **Catching False Negatives** - Honeypots can easily identify and capture new attacks or activity against them. Any activity with the honeypot is an anomaly, making new or unseen attacks easily stand out. This has been repeatedly demonstrated by the [Honeynet Project](#).
- **Minimal Resources** - Honeypots require minimal resources, even on the largest of networks. A simple Pentium computer can monitor literally millions of IP addresses on an OC-12 network.
- **Encryption** - It does not matter if an attack is encrypted, the honeypot will capture the activity.
- **IPv6** - It does not matter which IP protocol an attacker uses, honeypots will detect, capture and log all IP activity. In one documented case, a Solaris honeypot detected and captured an attack where attackers attempted to hide their communications using IPv6 tunneling within IPv4. On the other hand, there are almost no NIDS technologies that

can decode IPv6 or IPv6-tunneled traffic.

Combined, what this means is that honeypots can make extremely simple, cost-effective detection. By simple, the concept of honeypots is very easy to understand and implement. Just stick the box out there, if anything comes your way, you most likely caught yourself a bad guy. Honeypots have no rules to update or modify, and no advanced algorithms are required to analyze network traffic. Simplicity has its own inherent advantages. Honeypots are also an extremely cost effective solutions. By requiring little in the way of resources, maintenance, and analysis, they can dramatically reduce costs. Users do not need to spend a lot of money on a high-end box, that simple Pentium computer with the 10/100 network interface card will do fine on a OC-12 network. Even greater savings can be realized in manpower. Honeypots dramatically reduce the amount of information you have to collect, correlate, and archive. This reduction in man-hours allows security personnel to focus on other critical activities, such as patching. Honeypots also address some of the inherent failings of NIDS, such as detecting new attacks, or working in environments with encryption or IPv6 protocols.

That said, it must be stated that, like all technologies, honeypots have their disadvantages. Their greatest disadvantage is they have a limited field of view: they can only captured activity directed against them. Honeypots will miss attacks against other systems. If your Web server is compromised, your honeypot will not know it (unless the attacker uses the Web server to scan your honeypot). As such, I do not recommend using honeypots to replace any of your existing detection technologies. Instead, I recommend deploying them to work with your existing technologies to augment your current security strategy, as they can simply and cost effectively address many of the disadvantages of current technologies. Also, some people have questioned the legal issues of honeypots, specifically privacy. However, these issues apply mainly to research honeypots, which capture extensive amounts of information. Most production honeypots used for detection capture no more information then traditional security technologies, such as firewalls, routers, or NIDS. (A discussion on the legalities of honeypots is beyond the scope of this paper; however, a forthcoming installment in this series will discuss legal issues in greater depth.)

Conclusion

Honeypots are a powerful solution for detection. While a great deal of attention has been given to high-interaction, research honeypots, such as [honeynets](#), very little focus has been given to the detection capabilities of production honeypots. They address many of the disadvantages of

traditional detection capabilities. In our next article we look at an emerging issue with honeypots: legal issues.

To learn more about honeypots, check out <http://www.tracking-hackers.com>

Relevant Links

[SecurityFocus Honeypot Mailing List](#)

[Open Source Honeypots, Part One: Learning with Honeyd](#)

Lance Spitzner, SecurityFocus

[Open Source Honeypots, Part Two: Deploying Honeyd in the Wild](#)

Lance Spitzner, SecurityFocus

[Specter: A Commercial Honeypot Solution for Windows](#)

Lance Spitzner, SecurityFocus

[The HoneyNet Project](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus